Nondecomposable Data Dependent Regularizers offer Significant Performance Gains

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Data dependent regularization is known to benefit a wide variety of problems in machine learning. Often, these regularizers cannot be easily decomposed into a sum over a finite number of terms, e.g., a sum over individual example-wise terms. The F_β measure, Area under the ROC curve (AUCROC) and Precision at a fixed recall (P@R) are some prominent examples that are used in many applications. We find that for most medium to large sized datasets, scalability issues severely limit our ability in leveraging the benefits of such regularizers. Importantly, the key technical impediment despite some recent progress is that, such objectives remain difficult to optimize via backpropapagation procedures. While an efficient general-purpose strategy for this problem still remains elusive, in this paper, we show that for many data-dependent nondecomposable regularizers that are relevant in applications, sizable gains in efficiency are possible with minimal code-level changes; in other words, no specialized tools or numerical schemes are needed. Our procedure involves a reparameterization followed by a partial dualization – this leads to a formulation that has provably cheap projection operators. We present a detailed analysis of runtime and convergence properties of our algorithm. On the experimental side, we show that a direct use of our scheme significantly improves the state of the art IOU measures reported for MSCOCO Stuff segmentation dataset.

1 Introduction

Data dependent regularization is a mature and effective strategy for many problems in machine learning. In semi-supervised learning [1], the marginal distribution of the examples may serve to influence the estimation of the conditional distribution and in support vector machines, one could optimize the maximum *relative* margin based on the data distribution, rather than the *absolute* margin. In modern deep learning, data-dependent regularization is routinely used in both explicit and implict ways. A regularizer can take the form of certain geometric or clustering-type constraints [2, 3] on the representations that are learned by the network – for instance, using the distribution overlap between different classes [4] or seeking decorrelated autoencoder latent codes [5]. On the other hand, artificial corruption of the data is also a form of regularization – the widely used dropout method induces a regularization based on the Fisher information matrix of the samples (or its representations) [6]. More recently, the results in [7] offer a nice treatment of the form of data-dependent regularization that emerges from popular methods such as batch normalization and AdaGrad.

From Decomposable to Nondecomposable data-dependent regularizers. A number of data-dependent regularizers described above can be written as a sum of individual example-wise estimates of the regularizer. This setting is desirable because in order to run a SGD type algorithm, we simply pick a random sample to get an unbiased estimate of the gradient. But a number of application domains often necessitate a regularization criteria that may not nicely decompose in this manner. In such settings, a random sample of the dataset (or even a minibatch) does not necessarily provide us an unbiased gradient – biased gradients are known to adversely impact both the qualitative and quantitative performance of the training procedure, see [8].

Why are Nondecomposable regularizers relevant? Consider the situation where we would like to ensure that the performance of a statistical model is uniformly good over groups induced via certain protected attributes (such as race or gender). Or alternatively, we want that when updating an algorithm in a manufacturing process, the new system's behavior should mostly remain similar with respect to some global measures such as makespan [9]. And finally, when pooling datasets from multiple sites, global characteristics of Precision-Recall should be (approximately) preserved across sites [10]. Motivated by these issues encountered in various real world problems, recently [11] presents a comprehensive study of the computational aspects of learning problems with *rate constraints* – there, the constrained setting is preferred due to several reasons including its ease of use for a practitioner. The authors show that for a general class of constraints, a proxy-Lagrangian based method must be used because the Lagrangian is not optimal. This raises the question whether there exist a broad class of data-dependent *nondecomposable* functions for which the regularized/penalized formulation based on *standard* Lagrangian schemes may, in fact, be effective and sufficient. In this paper, we will address this question with simple examples shortly.

Examples in statistics. Nondecomposable regularization, in the most general sense, has also been studied from the statistical perspective, and is often referred to as *shape* constraints [12]. The need for shape constraints arise in clinical trials and cohort studies of various forms in the competing risk model, formalized using local smooth function theory [13]. While several authors have studied specific forms of this problem, the literature addressing the *computational* aspects of estimation schemes involving such regularizers is sparse, and even for simple objectives such as a sparse additive model, we find that results have appeared only recently [14]. Leveraging these ideas to train richer models of the forms that we often use in modern machine learning, establishing their convergence properties, and demonstrating their utility in real world applications is still an open problem.

Our Contributions. We first *reparameterize* a broad class of nondecomposable data-dependent regularizers into a form that can be efficiently optimized using first order methods. Interestingly, this reparameterization naturally leads to a Lagrangian based procedure where existing SGD based methods can be employed with little to no change. While recent results suggest that optimizing nondecomposable data-dependent regularizers may be challenging, our development shows that a sizable subclass of such regularizers indeed admit simple solution schemes. Our overall procedure comes with convergence rate guarantees and optimal per-iteration complexity. On the MSCOCO stuff segmentation dataset, we show that a direct use of this technique yields significant improvements to the state of the art, yielding a mean IoU of 0.32. These results have been submitted to the leaderboard.

2 Empirical Risk + Nondecomposable Data-dependent Regularizer

We setup some basic notations and concepts and then describe a simple model.

Basic notations. We assume that the training data is given as pairs of $(x, y) \sim \mathcal{D}$ where the joint distribution $\mathcal{D}$ is unknown. Here, x, y are random variables that represent examples (e.g., images) and labels respectively. We make no assumptions on x and y in that the marginal distributions of x and y can be discrete or continuous. Our goal is to estimate a function $f : x \mapsto y$ that achieves the minimum error measured using a specified loss function on the empirical/observed samples. We will use $W = \{w_1, w_2, \ldots, w_{l+1}\}$ to represent the trainable parameters of a feedforward neural network with $l + 1$ layers, with nonlinear activation functions. The output $W(x)$ of the function computed by the neural network W may be used to construct the classification or regression function f.

Nondecomposability. Now, suppose there exists a function φ such that $\varphi \circ \mathcal{D} =: s$ is a random variable called "*shape*" in a similar sense as described above [12]. Our notation is suggestive of the fact that s is **nondecomposable**, i.e., the value s_i for individual datapoint/example i may depend on the *entire* (empirical) distribution of $\mathcal{D}$. Moreover, this implies that without any further assumptions, *any* loss function $\mathcal{R}(\cdot)$ on $W(x)$ used to learn a model g to predict s cannot be represented as a decomposable function with respect to the training data x, y, see [15, 16].

As briefly noted in Section 1, nondecomposability poses unique challenges in designing algorithms for learning models with a large number of parameters. In most cases, existing backpropagation based techniques may *not* be directly applicable and require careful modifications [17]. In other cases, they may require significant computational overhead due to expensive projection operations [18]. Before we present our model, we now briefly describe two fundamental tasks that are, by and large, the most frequently used modules in vision based pipelines to see the benefits of nondecomposability.

Example 1. Object Detection seeks to simultaneously localize and classify objects in an image using a bounding box for each object [19]. Naturally, since the bounding boxes are rectangular, the *data-dependent* regularizer s is fully determined by the size of the objects. Here, we may use the knowledge of s to learn f that can perform equally well over *all* sizes present in the training data. But even though we may simply use class activation maps to compute the empirical distribution of s [20], the loss functions $\mathcal{R}(s)$ that are commonly used are **nondecomposable** as we will see shortly.

Example 2. Semantic Segmentation seeks to assign *each* pixel of an image to a class [21]. Recent works suggest that in order to train a model f, we may choose a model whose complexity strongly depends on the number of pixels in the images, see [22] for a recent survey. Unfortunately, these methods use up/down-sampling and so the learned representations do not directly offer immunity to variations in how much of the image is occupied by each class (i.e., size).

Remark 1. *We use these two examples to illustrate the applicability of our developments – since the use case corresponds to size, we may assume that the "shape" random variable is* discrete. *To that end, we will use $\mathcal{S}$ to denote the* countable scoring set *or the support of s with $|\mathcal{S}| < \infty$.*

2.1 An Optimization Model incorporating $\mathcal{S}-$measures

Let us write down a formulation which incorporates our shape regularizer. The objective function is a sum of two terms: **(1)** a *decomposable* Empirical Risk Minimization (ERM) term to learn the optimal function which maps x to y and **(2)** a *nondecomposable, data dependent* $\mathcal{S}-$measure regularizer term for s from x. In particular, for a fixed $\alpha > 0$, we seek to solve the following optimization problem,

$$\min_{W} \quad \overbrace{\frac{1}{N}\sum_{i=1}^{N} \text{loss}\left(W; x_i, y_i\right)}^{\text{ERM}} \; + \; \alpha \overbrace{\mathcal{R}\left(W; \varphi \circ \hat{D}\right)}^{\mathcal{S}-\text{Measure}}, \tag{1}$$

where $\hat{D}$ represents the empirical distribution, $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^k$ with $i = 1, \ldots, N$ denoting training data examples. We let loss$(\cdot)$ to be any standard loss function such as a cross entropy loss, hinge loss and so on. To keep the presentation and notations simple but nonetheless illustrate our main algorithmic developments, **we will assume that the $\mathcal{S}-$Measure is given by F_β metric** [23] while noting that the results apply directly to other measures such as R@P, P@R and can be easily generalized to other nondecomposable metrics such as AUCROC, AUCPR, using the Nyström method with no additional computational overhead [24].

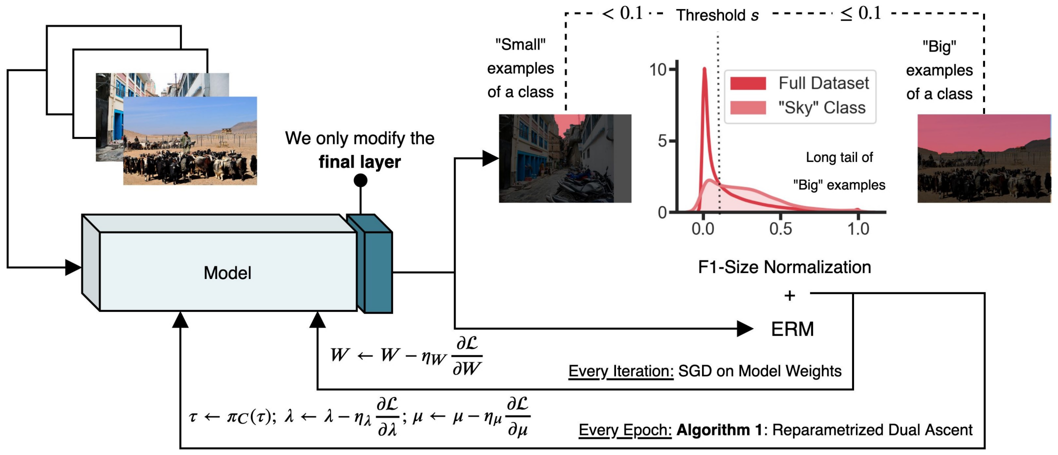
Remark 2. *For simplicity, we will suppress the dependence of f computed by the parameters W in our objective function* (1)*. For example, if f is represented using a deep neural network with weights W, then both terms in* (1) *may be nonconvex.*

3 Reparameterizing $\mathcal{S}-$Measures: Algorithm and Analysis

Since any backpropagation based procedure can easily handle decomposable terms, let us ignore the ERM term in our objective function (1) for the moment, and focus on the second term. Specifically, in this section, we will focus on the F_β metric as a placeholder for $\mathcal{S}-$measures. and show that there is a reformulation which will enable a backpropagation based procedure to solve our problem (1). For simplicity, let us assume that $s \in \{+1, -1\}$ and $\beta = 1$. The supplement shows how to extend our approach to any finite $|\mathcal{S}|$ and $\beta \in (0, 1]$ noting that in various applications in Section 4, $|\mathcal{S}| = 2$ suffices and is already quite effective (e.g., the learned model works whether s_i is $+1$ or -1).

The starting point of our technical development is the following result,

Observation 1 (Restated from [24])**.** *If f is linear, i.e., $y = W \cdot x$, we can represent the $\mathcal{S}-$measure term as a Linear Program (LP) as shown in equation (13) in [24].*



< 0.1
Threshold s
≤ 0.1
"Small" examples of a class
"Big" examples of a class
We only modify the **final layer**
Model
Full Dataset
"Sky" Class
Long tail of "Big" examples
F1-Size Normalization
+
ERM
$W \leftarrow W - \eta_W \frac{\partial \mathcal{L}}{\partial W}$
Every Iteration: SGD on Model Weights
$\tau \leftarrow \pi_C(\tau);\ \lambda \leftarrow \lambda - \eta_\lambda \frac{\partial \mathcal{L}}{\partial \lambda};\ \mu \leftarrow \mu - \eta_\mu \frac{\partial \mathcal{L}}{\partial \mu}$
Every Epoch: **Algorithm 1**: Reparametrized Dual Ascent

Figure 1: Our proposed training pipeline for optimizing nondecomposable $\mathcal{S}-$Measures.

Moving forward from Observation 1. In principle, one could use standard LP solvers to solve the LP in [24]. But this requires instantiating the constraint matrix (linear in the number of samples). This is impractical – the alternative in [24] is to use an iterative ascent method. In general this will also require a projection scheme, e.g., solving a QP. While more attractive that using an off-the-shelf method, both options may need cubic time. Further, this discussion of the computational issues only pertains to a *linear* objective – the setting expectedly becomes much more challenging for non-convex objectives common in deep learning architectures.

Problem (2): Slightly adapted form of LP in [24]

$$\min_{\phi,\tau,w,\epsilon} \quad n\epsilon + \sum_{i\in S^+} \tau_i + \sum_{i\in S^-} \phi_i \tag{2a}$$

$$\text{s.t.} \quad \tau_i \leq \epsilon, \tau_i \leq w_{l+1}\cdot a_l(x_i);\ \forall\ i \in S^+ \tag{2b}$$

$$\phi_i \geq 0\,, \phi_i \geq \epsilon + w_{l+1}\cdot a_l(x_i);\ \forall i \in S^- \tag{2c}$$

$$\sum_{i\in S^+} \tau_i = 1 \quad \epsilon \geq 0\,. \tag{2d}$$

3.1 Simplify and Reparameterize

We will first generalize the construction in [24] to be more amenable to non-linear functions that one often estimates with DNN architectures. Note that since our data-dependent $\mathcal{S}-$measures are specified on the input distribution of examples or representations derived from the transformations induced by the network W, we may denote representations in general as $a_l(x_i)$: in other words, the $l-$th layer provides us a *representation* of example i. We define $S^+ := \{i : s_i = +1\}$ (and similarly S^-) with $|S^+| = n$, and $n + |S^-| = N$, calculated using only the training data samples $X := \{x_i\}, Y := \{y_i\}, i = 1, ..., N$. We will still focus on the nondecomposable term but bring in the DNN loss when we describe the full algorithm.

Problem 3: a simplified LP

$$\min_{\tau,w,\epsilon} \quad n\epsilon + \sum_{i\in S^-} \max(0, \epsilon + w_{l+1}\cdot a_l(x_i)) \tag{3a}$$

$$\text{s.t.} \quad \tau_i \leq \epsilon, \tau_i \leq w_{l+1}\cdot a_l(x_i);\forall\ i \in S^+ \tag{3b}$$

$$\sum_{i\in S^+} \tau_i = 1 \quad \epsilon \geq 0\,. \tag{3c}$$

Eliminating redundancy in Problem (2). First, notice that we can eliminate the $\sum_{i\in S^+}\tau_i$ term from the objective function in (2a) since any feasible solution has to satisfy the first constraint in (2d). Second, using the definition of hinge loss, we can eliminate the ϕ_i terms in the objective by replacing ϕ_i with $\max(0, \epsilon + w_{l+1}\cdot a_l(x_i))$. This rearrangement leads to a LP in Problem (3).

Reparameterization Via Partial Dualization. Problem (3) has a objective which is expressed as a sum over finite number of terms. But overall, the problem remains difficult because the constraints (3b) are nonlinear and problematic. It turns out that a *partial dualization* – for only the problematic constraints – leads to a model with desirable properties. Let us dualize *only* the $\tau_i \leq w_{l+1}\cdot a_l(x_i)$ constraints in (3b) using λ_i, and the equality constraint in (3c) using μ, where λ_i and μ are the dual variables. This yields the Lagrangian,

$$L := n\epsilon + \sum_{i\in S^-} \max(0, \epsilon + w_{l+1}\cdot a_l(x_i)) + \mu\left(\sum_{i\in S^+}\tau_i - 1\right) + \sum_{i\in S^+}\lambda_i\left(\tau_i - w_{l+1}\cdot a_l(x_i)\right). \tag{4}$$

We can denote the set $C := \{(\tau_1,\tau_2,...,\tau_n,\epsilon) : \tau_i \le \epsilon, \epsilon \ge 0, i = 1,...,n\}$. With this notation, our final optimization problem for the $\mathcal{S}-$measure can be equivalently written as,

$$\max_{\lambda\ge 0,\mu}\ \min_{(\tau,\epsilon)\in C, W} L(\tau, w, \epsilon, \lambda, \mu). \tag{5}$$

We will shortly describe some nice properties of this model.

Relation to "Disintegration". While procedurally the derivation above is based on numerical optimization, the form in (5) is related to *disintegration*. Disintegration is a measure theoretic tool that allows us to rigorously define conditional probabilities [25]. As an application, using this technique, we can represent any (also, nondecomposable) measure defined over the graph of a convex function using (conditional) measures over its *faces* [26]. Intuitively, at the optimal solution of (5), the dual multipliers λ_i can be thought of a representation of the "disintegrated" $\mathcal{S}-$measure, as seen in the last term in (4). But in general, Lagrangians are difficult to use for disintegration because λ_i's need not sum to one; and more importantly, when the optimal solution is in the *interior* of the feasible set in Prob. (3), then $\lambda_i = 0\,\forall i$ by complementarity slackness. This means that the decomposition provided by the λ_i's need *not* be a nontrivial (probability) measure in a rigorous sense.

3.2 Core benefit of Reparameterization: Efficient Projections will become possible

The previous section omitted description of any real benefits of our reparameterization. We now describe the key computational benefits.

Can we perform fast Projections? Yes, via Adaptive Thresholding. Observe that for the formulation in (5) to offer any meaningful advantage, we should be able to show that the Lagrangian can indeed be efficiently minimized with respect to τ, ϵ, W. This will crucially depend on whether we can project on to the set C defined above. We now discuss that this holds for different batch sizes.

Fast projections when batch size, B is 2. Let us denote by B, the minibatch size used to solve (5) and consider the setting where we have S^+ and S^- as used in Probs. (2)-(3). In this case, we can randomly draw a sample i from S^+ and another i' from S^-. Observe that only one coordinate of τ, namely τ_i, changes after the primal update. Hence, the projection Π_C is given by the following simple rule: after the primal update, if $\epsilon < 0$, we simply set $\epsilon = 0$ leaving τ_i unchanged; on the other hand if $\epsilon > 0$, then we have **two** cases: **Case (1)** If $\tau_i \le \epsilon$, then do nothing, and **Case (2)** If $\tau_i \ge \epsilon$, then set $\tau_i = \epsilon = (\tau_i + \epsilon)/2$. In other words, the projection is extremely easy.

Remark 3. *We can indeed use multiple samples (and take the average) to compute gradients of weights W, ϵ, but only update one τ_i with this scheme.*

Generalizing the projection algorithm to larger batch sizes: $B > 2$. Note that projections onto sets defined by linear inequality constraints can be performed using standard (convex) quadratic programming solvers. But in the worst case, such solvers have a time complexity of $O(n^3)$. It is not practical to use such solvers efficiently in end-to-end training of widely used neural network architectures. It turns out that a much simpler scheme will work well.

Define $\boldsymbol{\tau} := [\epsilon; \tau] \in \mathbb{R}^{n+1}$, and $\boldsymbol{\gamma} := [\delta; \gamma] \in \mathbb{R}^{n+1}$. The Euclidean projection amounts to solving the following quadratic program:

$$\Pi_C(\boldsymbol{\tau}) = \arg\min_{\boldsymbol{\tau}} \|\boldsymbol{\tau} - \boldsymbol{\gamma}\|_2^2 \quad s.t. \quad \boldsymbol{\tau} \in C. \tag{6}$$

Our projection algorithm to solve (6) is given in Algorithm 2 which requires *only* a sorting oracle. The following lemma shows the correctness of Algorithm 2.

Lemma 1. *There exists an $O(B \log B)$ algorithm to solve* (6) *that requires* **only sorting and thresholding** *operations.*

Algorithm 1 Reparameterized Dual Ascent for solving (5)

```
1: Input: $X, Y, S$, trainable parameters: $W, \tau$
2: Initialize variables $W, \tau, T$, Epochs, $\eta_d$
3: for $e = 1, \dots,$ Epochs do
4:     for $t = 1, \dots, T$ do
5:         SGD on ERM in (1) + Lagrangian (4)
6:     end for
7:     $\tau \leftarrow \Pi_C(\tau)$ using Algorithm 2
8:     $\lambda \leftarrow \lambda + \eta_d(\tau - w_{l+1} \cdot a_l(x_i))$
9:     $\mu \leftarrow \mu + \eta_d(1^T\tau - 1)$
10: end for
11: Output
```

Algorithm 2 Projection operator: $\Pi_C(\tau)$

```
1: Input: $\tau, B > 2$
2: Output: $\Pi_C(\tau)$
3: If $\tau_1 < 0$, set $\tau_1 = 0$.
4: Compute largest $k^*$ such that
5: $\tau_{k^*+1} \le \frac{1}{k^*}\sum_{i=1}^{k^*}\tau_i$ by sorting $\{\tau_i : i > 1\}$.
6: Output: Set $\bar{o} = \frac{1}{k^*}\sum_{i=0}^{k^*}\tau_i$,
7: Return $\Pi_C(\tau_i)$ where
       $\Pi_C(\tau_i) = \bar{o}$ for $i \le k^*$ or        (11a)
       $\Pi_C(\tau_i) = \tau_i$ for $i > k^*$.            (11b)
```

Proof. WLOG, we can assume that $\gamma_2 \ge \gamma_3 \ge \cdots \ge \gamma_{n+1}$ since when given any γ, we may rearrange the coordinates. We may also assume that for any $i > 1$, $\gamma_i \ne \gamma_{i+1}$ by picking an arbitrary order for coordinates with the same value. The first order optimality conditions for (6) are given by:

$$\text{(Feasibility)} \quad \text{Primal: } \tau_1 \ge \tau_i \,\forall\, i \in \{2, \dots, n+1\}, \quad \tau_1 \ge 0, \;\; \text{Dual: } \mathsf{d} \ge 0, \tag{7}$$

$$\text{(KKT Conditions)} \quad \tau_i - \gamma_i + \mathsf{d}_i = 0 \,\forall\, i \in \{2, \dots, n+1\}, \;\; \tau_1 - \gamma_1 - \mathsf{d}_1 - \sum_{i=2}^{n+1}\mathsf{d}_i = 0, \tag{8}$$

$$\text{(Complementarity Slackness)} \quad \mathsf{d}_i(\tau_i - \tau_1) = 0 \,\forall\, i \in \{2, \dots, n+1\}, \;\; \mathsf{d}_1\tau_1 = 0, \tag{9}$$

where d represents the vector containing the Lagrange multipliers for the constraints C in (6). From (8), we have that $\mathsf{d}_i = \gamma_i - \tau_i$ which implies that and $\tau_1 = \gamma_1 + \sum_{i=1}^{n+1}\mathsf{d}_i \ge \gamma_1$. From (9), either $\mathsf{d}_i = 0$ or $\tau_i = \tau_1$, and $\mathsf{d}_1 = 0$. Let $I := \{i \ne 1 : \tau_i = \tau_1\}$ be the (active) set of indices at the optimal solution. Given I, we can easily find τ_1 by solving the following 1D quadratic problem

$$\min_{\tau_1 \ge 0} \sum_{i \in I \cup \{1\}} \frac{1}{2}(\tau_1 - \gamma_i)^2 \implies \tau_1 = \frac{1}{|I| + 1}\sum_{i \in I \cup \{1\}} \gamma_i. \tag{10}$$

So, it is now clear that in order to satisfy the feasibility conditions (7), we simply choose the first $k+1$ coordinates of γ_1 to be the index set I, and set τ_1 to be the average of $\{\gamma_i\}$ where $i \in I$. □

3.3 Collecting the results to obtain a numerical optimization scheme

With the results in the preceding section, we now describe below an efficient approach to solve (1).

Reparameterized Dual Ascent. Using the projection operator above (Alg. 2), our final algorithm to solve $\mathcal{S}-$Measures regularized problem (1) is given in Algorithm 1 where we use η_d to denote the dual step size. The following theorem characterizes the convergence rate of Algorithm 1.

Theorem 2. *Assume that $\|\nabla_W loss(W; x_i, y_i)\|_2 \le G_1$, and Var$_i$ $(\|\nabla_W loss(W; x_i, y_i)\|_2) \le \sigma$ in the ERM term in Prob.* (1). *Then, Alg. 1 converges to a $\epsilon-$approximate (local) solution of Prob.* (1) *in $O(1/\sqrt{T})$ iterations.*

Proof. We will use Theorem 1 in [27] with $\rho = G_2 = \sqrt{d}$ and $C_2 = d$, see supplement. □

Remark 4. *(Implementation details of Alg. 2.) In the supplement, based on Lemma 1, we describe a one pass scheme to compute k^* in Step 4 of Alg. 2. So, the time complexity of Alg. 2 is $O(B \log B)$.*

Discussion. The reader will notice that the projection step in Algorithm 1 is *outside* the inner loop, whereas classical Lagrangian based methods guarantee convergence when the projection is performed in the *inner* loop. One advantage of having the projection outside the inner `for` loop is that SGD type methods allow for faster optimization of the Lagrangian with respect to the primal variables W. That is, it is now known that constant step size policies guarantee faster convergence of the inner `for` loop under some structural assumptions [28]. In any case, we give a detailed description of our

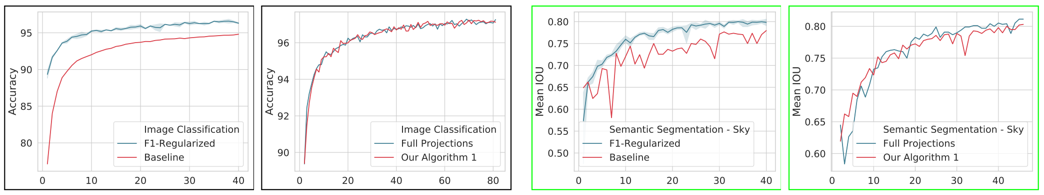


Figure 2: (From Left) Performance of our Reparameterization on $\mathcal{S}-$MNIST (Cols. 1&2) and MS-COCO class "Sky" (Cols. 3&4) datasets: $\mathbf{x}$-axis denotes the Epoch in all the plots. Observe that across both the datasets, Reparameterized Dual Ascent Algorithm (red) 1: (i) (Cols. 1&3) obtains high quality solutions than the Baseline (blue) [29]; (Cols. 2&4) and requires significantly less projections (once every epoch from Theorem 2).

Algorithm 1 in the supplement with a discussion of the trade-off between using projections in the inner `for` loop versus the outer `for` loop in Algorithm 1.

Computational Complexity. Assume that each gradient computation is $O(1)$, Alg. 1 requires $T = O(1/\epsilon^2)$ iterations to compute a local ϵ-optimal solution (inner loop). We use a constant number of epochs E (outer loop), so the total cost is $O(E(1/\epsilon^2 + n\log n))$. Details are in the supplement.

4 Experimental Evaluations

Overview. We show experiments on three different tasks: $\mathcal{S}-$Measures defined using the F_1 metric can be efficiently optimized using our Reparameterized Dual Ascent Algorithm 1 in large scale vision problems while strongly improving the empirical performance. The **first** set of experiments is designed to show that the F_1 metric based objectives may benefit existing classification models using convex classifiers. The **second** set of experiments further evaluates the performance of Alg. 1 for nonconvex models. Our goal here is to show that nondecomposable regularizers *can* be used to stabilize models without sacrificing on the overall accuracy. In the **third** set of experiments, we show that architectures used for Semantic Segmentation can be trained using Alg. 1. In all our experiments, we used ADAM optimizer (with default parameters) to train with primal step sizes of $\eta_\tau = \eta_\epsilon = \eta_W = 10^{-2}$, and dual step sizes of $\eta_\lambda = 10^{-3}$, and $\eta_\mu = 10^{-5}$. We report the average of results over the regularization parameter $\alpha = \{10^{-2}, 10^{-3}, 10^{-4}\}$. Baseline corresponds to $\alpha = 0$.

Experiment 1) Improving Linear Classifiers using $\mathcal{S}-$Measures **Dataset.** Consider a dataset with distinct classes: MNIST with 10 classes (digits). In MNIST, the *size* of the foreground (pixels with nonzero intensities) is about the same for all images (and each class) in MNIST – the distribution of foreground size s' is *unimodal* (see supplement). To see the benefits of $\mathcal{S}-$measures, we create an augmented dataset called $\mathcal{S}-$MNIST. We added a set of images in which the foreground is double the original size in MNIST simply by rescaling the foreground to get a **two mode** s'.

How to obtain size s? To get $s \in \{0, 1\}$, we can simply threshold s' using the empirical mean.

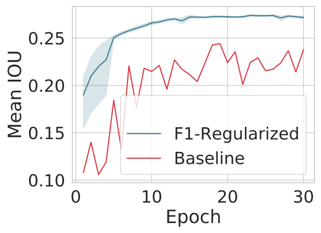
Model and Training. We used a single layer neural network f to predict y from x but added an extra node to account for the $\mathcal{S}-$Measure term in (1). That is, our total loss is the sum of the softmax cross-entropy ERM loss and the $\mathcal{S}-$Measure (using F_1 metric). We trained using Alg. 1 for 40 epochs with a minibatch size of 100.

Results. We see the benefits provided by the F_1 metric as a data-dependent regularizer in Fig. 2. Column 1 shows that our model is uniformly more accurate than the baseline throughout the training process. Moreover, Column 2 compares Alg. 1 with classical dual ascent procedures from [11]. Here, full projections refers to computing $\Pi_C(\cdot)$ after every inner iteration in Alg. 1. Clearly, we can see that Algorithm 1 obtains high quality solutions but needs **one projection operation** every epoch.

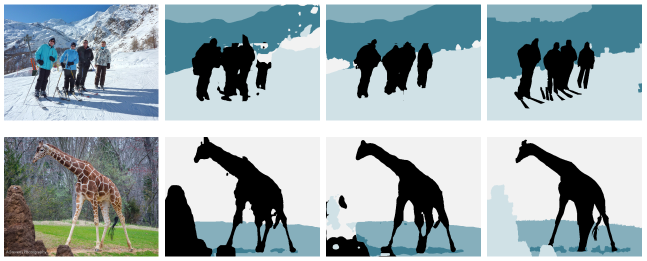
Takeaway #1. Models obtained using Alg. 1 are more accurate and stable for linear classifiers.

Experiment 2) Improving one class Segmentation using $\mathcal{S}-$Measures **Dataset.** We consider the task of the pixel-wise contextual labeling of an image. We found that the "sky" category in the *MSCOCO stuff* (2017) dataset [30] has a high variance over the samples in terms of size: so, taking this property into account seems sensible. So, we use **only** the images in the "sky" category [30].

How to obtain size s? We first computed the empirical distribution of number of pixels s' that are labeled as sky in the training set. We then pick the mean of s' to be the threshold to obtain a binary s.



(a) We outperform the baseline on MSCOCO 164K while being stable.



(b) Colors (except black) indicate different “stuff” categories in an image. From left to right: original image, baseline, our result and ground truth.

Figure 3: Quantitative/qualitative results on MSCOCO stuff segmentation benchmark. Our mean IOU improves upon the state of the art reported in [21] by $\approx 10\%$. Results have been submitted to the leaderboard.

Model and Training. We utilize SegNet [31, 32], a deep encoder-decoder architecture for semantic segmentation for its simple design and ease of use. As before, we add a fully-connected layer at the end of the model to incorporate the $\mathcal{S}-$Measure specified by F_1 metric. We trained our models for 40 epochs with a learning rate of 0.01 and batch size 7.

Results. Figure 2 Column 3 shows our results averaged over the thresholds $\{0.55, 0.6, 0.65\}$. We can see that our models uniformly outperforms the baseline with a 80% mean IOU (over 77.6% baseline). Furthermore, observe that our Alg. 1 is much more stable than the baseline throughout the training process even in nonconvex settings while requiring the fewest projection steps.

Takeaway #2. Performance boosts from the F1-regularizer carries over from the simple convex classification task to the pixel-wise (nonconvex) classification task with a deep neural network.

Experiment 3) Improving Semantic Segmentation with Nondecomposable Regularizers Dataset. We are now ready to consider semantic segmentation with multiple classes in each image. Our dataset consists of $164K$ images that belong to any of the “stuff” classes in the MSCOCO “Stuff” dataset [30]. We downsampled the training images to 106×160 size to reduce training time.

How to obtain size s? The *volume of stuff* c denoted by s'^c is measured by the number of pixels that belong to c. We observed that s'^c is close to 0% for most c (see supplement). So, we picked a threshold of ≈ 0.05 to obtain a binary $s^c \in \{0, 1\}$ for each class c. Then, we use the majority vote provided by all classes present in an image to obtain $s \in \{0, 1\}$ for individual images – corresponding to “big/small”. That is, if the majority of the classes present inside the image are “big” (as determined by the threshold $s = 0.05$), then we assigned the image to be “big” and vice-versa.

Model and Training. We use DeepLabV3+ model proposed in [21] for training. DeepLabV3+ is a popular model for semantic segmentation (needs no CRF post-processing) and can be trained end-to-end. We used a minibatch size of 144 for baseline and 120 for our F_1 regularized models.

Results. Figure 3 Column 1 shows the quantitative results of our experiment averaged over the thresholds $\{0.01, 0.03, 0.05, 0.07, 0.09\}$. Columns 2-4 shows some of our qualitative results.

Takeaway #3. On the MS COCO 164K Stuff dataset, we achieve state of the art results with 0.32 Mean IOU (versus 0.30 current state of the art mean IOU reported in [21]).

5 Conclusions

While nondecomposable data-dependent regularizers are variously beneficial and needed in a number of applications, their benefits cannot often be leveraged in large scale settings due to computational challenges. Further, the literature provides little guidance on mechanisms to utilize such regularizers within the deep neural network architectures that are commonly used in the community. In this paper, we showed how various nondecomposable regularizers may indeed permit highly efficient optimization schemes that can also directly benefit from the optimization routines implemented in mature software libraries used in vision and machine learning. We provide a technical analysis of the algorithm and show that the procedure yields state of the art performance for a semantic segmentation task, with only minimal changes in the optimization routine.

References

[1] A. Corduneanu and T. Jaakkola, “Data dependent regularization,” *Semi-Supervised Learning*, pp. 163–182, 2006.

[2] J. Lezama, Q. Qiu, P. Musé, and G. Sapiro, “Olé: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[3] W. Zhu, Q. Qiu, J. Huang, R. Calderbank, G. Sapiro, and I. Daubechies, “Ldmnet: Low dimensional manifold regularized neural networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev, “Metric learning with adaptive density discrimination,” *arXiv preprint arXiv:1511.05939*, 2015.

[5] B. Cheung, J. A. Livezey, A. K. Bansal, and B. A. Olshausen, “Discovering hidden factors of variation in deep networks,” *arXiv preprint arXiv:1412.6583*, 2014.

[6] S. Wager, S. Wang, and P. S. Liang, “Dropout training as adaptive regularization,” in *Advances in neural information processing systems*, 2013.

[7] W. Mou, Y. Zhou, J. Gao, and L. Wang, “Dropout training, data-dependent regularization, and generalization bounds,” in *International Conference on Machine Learning*, pp. 3642–3650, 2018.

[8] J. Chen and R. Luss, “Stochastic gradient descent with biased but consistent gradient estimators,” *arXiv preprint arXiv:1807.11880*, 2018.

[9] G. B. Limentani, M. C. Ringo, F. Ye, M. L. Bergquist, and E. O. McSorley, “Beyond the t-test: statistical equivalence testing,” 2005.

[10] H. H. Zhou, Y. Zhang, V. K. Ithapu, S. C. Johnson, V. Singh, *et al.*, “When can multi-site datasets be pooled for regression? hypothesis tests, ℓ_2-consistency and neuroscience applications,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 4170–4179, JMLR. org, 2017.

[11] A. Cotter, H. Jiang, S. Wang, T. Narayan, M. Gupta, S. You, and K. Sridharan, “Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals,” *arXiv preprint arXiv:1809.04198*, 2018.

[12] P. Groeneboom and G. Jongbloed, *Nonparametric Estimation under Shape Constraints: Estimators, Algorithms and Asymptotics*. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, 2014.

[13] L. Chenxi and J. Fine, “Smoothed nonparametric estimation for current status competing risks data,” *Biometrika*, 2012.

[14] J. Yin and Y. Yu, “Convex-constrained sparse additive modeling and its extensions,” *CoRR*, 2018.

[15] H. Narasimhan, R. Vaish, and S. Agarwal, “On the statistical consistency of plug-in classifiers for non-decomposable performance measures,” in *Advances in Neural Information Processing Systems*, 2014.

[16] A. Sanyal, P. Kumar, P. Kar, S. Chawla, and F. Sebastiani, “Optimizing non-decomposable measures with deep networks,” *Machine Learning*, 2018.

[17] X. Zhang, M. Liu, X. Zhou, and T. Yang, “Faster online learning of optimal threshold for consistent f-measure optimization,” in *Advances in Neural Information Processing Systems*, pp. 3889–3899, 2018.

[18] P. Kar, H. Narasimhan, and P. Jain, “Online and stochastic gradient methods for non-decomposable loss functions,” in *Advances in Neural Information Processing Systems*, 2014.

[19] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE transactions on neural networks and learning systems*, 2019.

[20] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[21] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 801–818, 2018.

[22] X. Liu, Z. Deng, and Y. Yang, "Recent progress in semantic image segmentation," *Artificial Intelligence Review*.

[23] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, "Optimal thresholding of classifiers to maximize f1 measure," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer.

[24] E. E. Eban, M. Schain, A. Mackey, A. Gordon, R. A. Saurous, and G. Elidan, "Scalable learning of non-decomposable objectives," *arXiv preprint arXiv:1608.04802*, 2016.

[25] A. M. Faden *et al.*, "The existence of regular conditional probabilities: necessary and sufficient conditions," *The Annals of Probability*, 1985.

[26] L. Caravenna and S. Daneri, "The disintegration of the lebesgue measure on the faces of a convex function," *Journal of Functional Analysis*, 2010.

[27] M. Mahdavi, T. Yang, R. Jin, S. Zhu, and J. Yi, "Stochastic gradient descent with only one projection," in *Advances in Neural Information Processing Systems*, 2012.

[28] A. Dieuleveut, A. Durmus, and F. Bach, "Bridging the gap between constant step size stochastic gradient descent and markov chains," *arXiv preprint arXiv:1707.06386*, 2017.

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[30] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*, IEEE, 2018.

[31] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[32] M. P. Shah, "Semantic segmentation architectures implemented in pytorch.," *https://github.com/meetshah1995/pytorch-semseg*, 2017.